International Journal of Scientific Engineering and Technology
Volume No.5 Issue No.12, pp: 537-541

ISSN:2277-1581
01 December 2016

Software Defined Network for Data Center Using Open Flow Protocol

Shamam A. Chasib, Dr. Abdulkareem A. Khadim
Al-Nahrain University, Baghdad, Iraq
shamamamer@yahoo.com, abdulkareem.a@coie.nahrainuniv.edu.iq

Abstract: ***The conventional Data Center Networks (DCNs) are highlighted by the increased number of users and bandwidth requirements which in turn have many implementation limitations. The current networking devices with its control and forwarding planes coupling resulted in network architectures that are not suitable for dynamic computing and storage needs. Software Defined networking (SDN) is introduced to change this notion of traditional networks by decoupling the control and the forwarding planes. In this paper a programmable data center network architecture using SDN based OpenFlow protocol is demonstrated to measure some of the network performance parameters such as delay and throughput and compared them to that of traditional network. In addition, the possibility of adding a Load Balancing application is also introduced for DCN using multi-POX controllers in the SDN network. Two scenarios of DCNs using tree topology are used to implement the traditional and SDN based networks. In the first, the delay performance of DCN is reduced in the SDN based network by about 15% in comparison to that of traditional network due to OpenFlow protocol and the centralized POX controller. The use of SDN slightly increases the throughput when compared to traditional network with about 1.7 Mbit/sec.. In the second scenario; a load balancing application is implemented in tree DCN/SDN network topology using two POX controllers to handle traffic distribution among many servers. This leads to better resource utilization as a result of using SDN.***

Keywords: DCN; SDN; Open Flow; POX; Load Balancing.

I. Introduction

In the last two decades, networks had been changed according to the rapid changing in its requirements. The current Data Center Networks have large numbers of hosts may be tens or thousands with special needs of bandwidth as the cloud network and the multimedia content computing are increased [1]. As a result, the existing technologies of DCNs have two problems: the first is the limitation imposed by the present DCNs physical infrastructure which is usually dedicated to a specific task and a limited amount of data handling capability. Thus any new equipment required massive efforts to install and monitor. Second, due to the rapid increase in the number of applications, the websites and storage space resources are being underutilized due to static routing mechanisms. To overcome these limitations, a Software Defined Network based Openflow Data Center network architecture [2] is used to obtain better performance and implement various network applications such as traffic load balancing.

As many researchers used Mininet as a software to implement SDN and to measure the performance of the controllers, In [3] the researchers covered the emulation of SDN using Mininet in different simulation environments, where many tests used to study Mininet limitations related to resource capabilities and simulation environment. They tested the scalability of Mininet in creating multiple topologies with different number of nodes. Their results show that there exists an effect of the simulation environment on the required time for building up the topology[3]. In addition, some attempts had implemented a multiple service load balancing using OpenFlow network [4]. Their approach integrated the network functionality of load balancing to reduce the efforts needed to maintain the network and increase the efficiency, where three NOX controllers were used in their topology with FlowVisor application. They concluded that a dedicated and specialized load balancing algorithms can be implemented with SDN network depending on the requirements of workloads and services. As a result the present work takes the aspect of load balancing and the use of multi controlling software to implement SDN based network.

II. Software Defined Networks

SDN is a notion which has lately gained the interest of researchers in the field of computer networking. It provides a way for programming the current networks and shifting the awareness of the networking community by simplifying the design and management compared to the existing approaches. The tight coupling between the network's controlling plane (the place where the traffic handling decisions are made) and the network forwarding plane (where the forwarding of the traffic takes place) causes the designing and managing process of the networks to become an intimidating task because of the high level of involved complexity [5]. Various challenges related to the evolution and management of the network arise throughout recent years. Network managers and developers had to transform the network policies of high level to low-level commands manually. This process can be very challenging for complex networks [6].

When a functionality is introduced as a new feature to the network, like load balancers [7], it usually needs manipulating with the existing infrastructure and standardization to ensure the ability of being interoperable among the various vendors implementations. In the SDN control software, which is considered as the brain of the whole architecture, is called the SDN OpenFlow (OF) Controller that has the entire view of the network and responsible of the decision making. On the other hand, hardware entities such as switches or routers are handling the forwarding process of the packets to the destination

ISSN:2277-1581
01 December 2016

according to the instructions provided by the controller. The splitting of the controlling layer from the underlying forwarding one simplifies network evolution [8]. Many protocols and network applications can be implemented and tested over the network without affecting the network traffic and many additional infrastructures can be introduced without much hassle.

The OF protocol is an open, and standard protocol that defines how the control plane in an SDN environment can be controlled and configured by a centralized controller. When using OF, the controllers have the capability to manage how the data packets are forwarded inside the network. The information stored in multiple formats (Mac tables, Routing tables) in the switches and the routers are calculated by many complex switching and routing protocols. The forwarding plane in the traditional networks is programmed based on these tables. OF protocol provides a standardized and centralized protocol that can manage all the flow tables through the use of controlling software that accommodate this protocol [9].

III. Data Center Networks

A DCN is a facility used to house servers and computer hosts which are connected using dedicated switches and links. Current Data Centers may have large number of hosts with specific bandwidth and delay requirements [10]. The evolutionary trends in the form of cloud computing, the increased density and the virtualization needs had placed critical networking requirements on data center world. Some of these are dominating the current network limitations which represented by the problem of the networking protocols which are coupled with the physical ports of network devices. Further, the nature of virtual machines had placed many requirements on the network's components capacity, such as bursting of network MAC addresses, number of virtual LANs, and the spanning tree [11].

In Traditional DCNs, a large number of routers and switches have been included in such type of network where the traffic is forwarded based on the limited view of each network device. This networking model is unable to meet the proper flexibility required with the current massive data amounts. In addition the strict coupling between the software and the hardware of each networking device results in an expensive scaling, and complexity to maintain [12]. To address these issues, organizations are turning to software-defined networking. With SDN, traffic flow is managed with software applications, which are significantly more dynamic and allow optimization. Scalability can be easily achieved in SDN, since the software scales to as many routers or switches as there are in the network [13].

Adding hardware creates new paths for the software to monitor, manage, and use to create the most functional traffic flow. With a solution such as SDN based network, the routing of the network can be customized much more easily, shaping it to the particular interests and needs of that DCN.

Whether a company is looking for cloud provisioning or high-performance computing from its DCN, SDN enables a network experience that will raise the speed and the flexibility in a way to encourage the innovation. SDN is vendor-neutral so that it can be suitable to any equipment type. As long as there exists SDN interface to the devices of the network, it does not matter to the applications which operating system is being used or who provided the hardware [12].

IV. Implementation

In order to implement and understand programmable networks and to accomplish the Data Center architecture, it is important to have an environment of several specifications. Tree topology is used in this work which combines both the star and bus topologies [14]. Fig. 1 shows the tree topology used in the work. It consists of three layers model; the first is the core layer which is responsible for connecting the network to the internet, the aggregation layer which interconnects the access layer switches with the core, and the third is the access layer representing the switches which connect the underlying hosts.

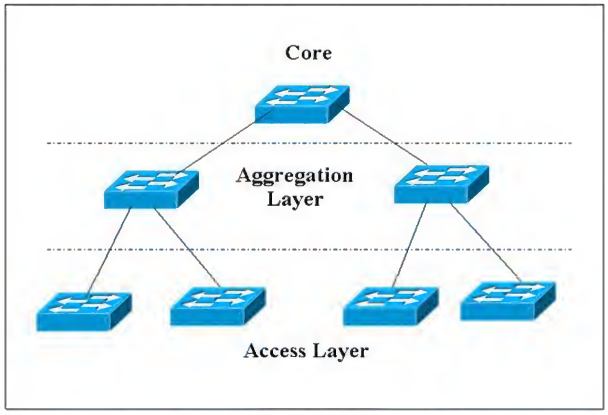

Fig.1:Tree Network Topology

Mininet [15] is a tool used for network emulation used to create a network of virtual switches, hosts, links, and controllers on one Linux kernel. Mininet's switches support OF protocol for Software-Defined Networking and highly adaptable custom routing. Mininet had provided many benefits to the SDN paradigm, such as providing an inexpensive and simple network framework for evolving applications of OpenFlow. Mininet enables the testing of topology, without wiring up the network physically and has a (CLI) for debugging network tests.

The software that is responsible for the behavior of the network and the implementation of the desired application is the OF controller. This software is the most important part of the work, due to its features and specifications, POX controller was chosen in the scenarios [16].

The design consists of two POX controllers that are directly connected to the network; the first is the controller (C0) which is responsible for programming the switches to forward the traffic as a layer-2 switch and is used in both scenarios[17]. The second controller (C1) is used only in the second scenario to enable the switch performing the load balancing function. The controller (C0) in both scenarios contains the module of the forwarding function which is basically permits the traffic forwarding between all of the switches resides in the network that are connected to this controller. The reason behind using this controller is to provide the control plane separation from all

ISSN:2277-1581
01 December 2016

the switches and to program the switches to be forwarding devices by exchanging the flow tables between them. The second controller (C1) operates the module of Load Balancing application that makes one of the switches (S4) in the second scenario acting as a Load Balancer. Each controller has the specification of supporting Python 2.7, OpenFlow 1.0, and connects with Open Virtual Switch (OVS) [18],[19],[20].

V. Evaluation

A. *First Scenario*

Fig.2 and Fig.3 show the traditional and the SDN based network respectively, both were used to implement the first scenario. The performance of both networks is measured through the use of **ping** tool to measure the delay and the **iperf** tool to measure the throughput.

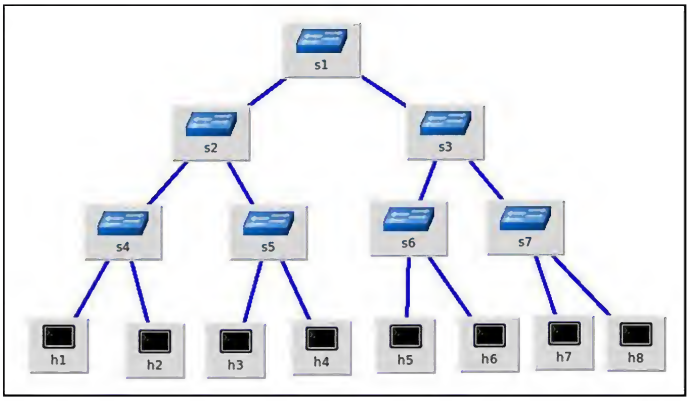

Fig.2 Traditional Tree Data Center Network

The delay and throughput are measured for the link connecting h1 and other hosts in the network. The delay is measured with different number of packets for each flow (here 100 and 500, and 1000 packets were used).

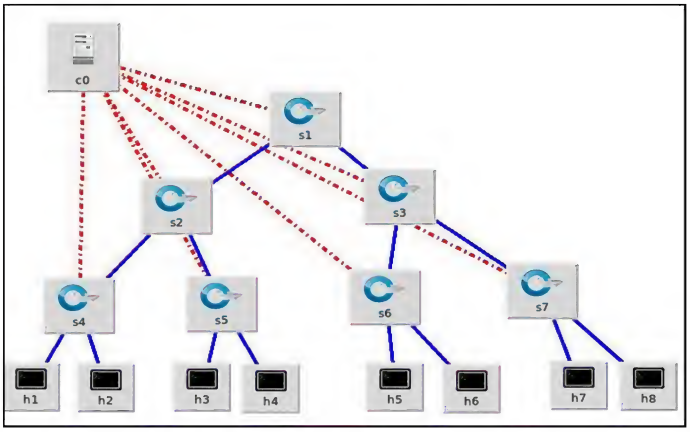

Fig. 3 SDN Tree Data Center Network

TABLE I. MEASURED DELAY WHEN TRANSMITTING 100 PACKETS

Network Type	Communicating Hosts	Delay(ms)
Traditional	From h3 to h1	3.1
	From h5 to h1	3.2
	From h7 to h1	4.1
SDN based	From h3 to h1	2.5
	From h5 to h1	2.6
	From h7 to h1	3.2

TABLE II. MEASURED DELAY WHEN TRANSMITTING 500 PACKETS

Network Type	Communicating Hosts	Delay(ms)
Traditional	From h3 to h1	2.5
	From h5 to h1	3.6
	From h7 to h1	3.4
SDN based	From h3 to h1	1.9
	From h5 to h1	2.9
	From h7 to h1	2.8

TABLE III. MEASURED DELAY WHEN TRANSMITTING 1000 PACKETS

Network Type	Communicating Hosts	Delay(ms)
Traditional	From h3 to h1	2.7
	From h5 to h1	3.5
	From h7 to h1	3.3
SDN based	From h3 to h1	2.2
	From h5 to h1	2.8
	From h7 to h1	3.0

From the above tables, it is clear that the delay in SDN based network had been reduced in all the three cases. This is due to two reasons; the first reason behind this is the delay caused by Routing and Switching, where in IP traditional networks, the IP packets are forwarded through a series of switches or routers. This will continuously update their decisions about which is the next best route to deliver the packet to its final destination. Thus the router, switch, and any congestion in the path contribute to the total delay. The second reason is the nature of data communication protocols at different layers in the protocol stack, where handshaking for synchronization between the transmitter and the receiver is needed in traditional network. All the handshakes take time to propagate across the link and can be added to the latency of the transmitting information from source to destination. This is unlike the SDN based network where the splitting of the controlling and the forwarding planes plays a big role in reducing this effect using OpenFlow protocol. This allows the ease of programming to all the switches in the network by the controller where all the operations can be handled using this controller. So, even if there exist a new flow (set of packets) transmitted, only the controller will be responsible for programming the given flow rule in the switches.

The second performance parameter is the throughput. Iperf tool is used to measure the throughput for this scenario. Iperf is used

ISSN:2277-1581
01 December 2016

as a network traffic generation tool that is developed for making measurements of the performance of TCP/UDP transmission. The throughput is measured for both traditional and SDN based network. The traditional network has a throughput of 58.2 Mbit/sec, while in SDN based network the throughput is 59.9 Mbit/sec. There is a little difference between both networks. The main reason is that SDN based network acts similarly as the traditional network as long as SDN is implemented in the proactive mode.

B. The Second Scenario

The implementation of the load balancer application is represented by a switch which act here as a traffic load balancer that distributes load among two HTTP servers. The network used here is the SDN based tree datacenter network that works with two controllers. The first has the functionality of making the switches forward traffic and the other letting one switch to act as load balancer. Fig.4 shows the SDN based data center network with two POX controllers.

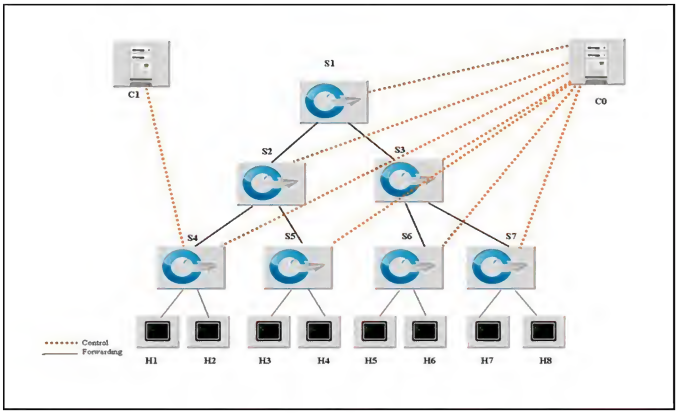

Fig. 4 SDN Application-Based Data Center Network

The evaluation tool used here is the **SimpleHTTPServer** which is a module that comes with Python and has a simple HTTP server action that provides standard GET and HEAD requests handler. An advantage with the built-in HTTP server is that there is no need to install and configure any additional tools such as Python interpreters because it comes by default with almost all Linux distributions. The only thing that is needed is to have Python programming language. The second tool is the **curl** which is a tool to transfer data from or to a server using one of the supporting protocols such as FTP, HTTP, FTPS, HTTPS,…etc. **Curl** displays this data to the terminal by default, so when the client requests a page using curl, the content will be directly displayed in the client's terminal.

Two hosts are considered as servers (h1and h2) in Fig.5 and another two hosts were picked as clients (h3 and h4). Any other host could also be taken in place of these. The test began with making the first two hosts as servers each of which running on port 80.

Fig. 5 Hosts h1 and h2 Acting As HTTP Servers

The clients were run by first installing the curl tool using the Linux command; $ sudo apt-get install curl

The curl command is issued then in each client with the service IP address 10.0.0.9 to request the web page from the servers. Thus in this step the switch that is connected to the servers would act as load balancer and started to distribute the load requested by each client to a specific server. The reason of doing this is to prevent making one of the servers being underutilized and all the other requests go to only one server. Instead, the requests will be distributed among both servers.

The complete load balancing is not achieved without the use of two SDN POX controllers, the reason behind this is a twofold; first: with one controller a problem of using multi-switch in the tree network is raised. If the request is made by the client to a server the controller is confused "which switch will be chosen to distribute the load". To solve this problem, a DPID could be defined in the module which is a number that is given to each switch and choose a specific switch to act as a load balancer. This method may cause errors in programmability; if one line of code is programmed with error then the whole module will not work.

Another method (used in this work) which make use of multi-controllers (two POX controllers) to prevent the problems that may happen in modifying the application's code and also to reduce the failure in the case of losing the centralized controller (where the forwarding and the load balancing functions reside). By using this method, one controller becomes responsible for programming the switches to forward the data, while the other controller acts as load balancer. Fig.6 shows how the requests are distributed by the switch (S4) to different servers.

After sending multiple replies from the servers to the clients, the http page will be displayed in the client's terminal. This process is shown in Fig.7 for the case where host h3 is chosen.

ISSN:2277-1581
01 December 2016

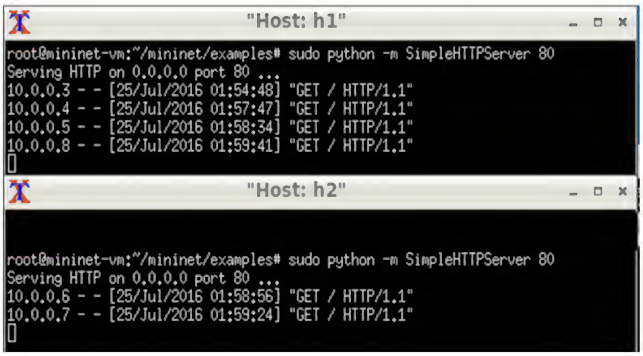

Fig. 6 HTTP Traffic Distribution Between h1 & h2

Fig. 7 HTTP The Host h3 After Receiving The Page From h1

VI. Conclusion

SDN becoming the promising network technology that supports the ever-growing today's network demands. It is shown that SDN is more flexible in achieving better network performance when compared to traditional network. This can be achieved without adding new equipment or manually programming all devices, the only requirement is to have a switch or router with an enabled feature of OpenFlow protocol.

For the test condition considered in the work, comparing SDN with traditional network showed an improved delay and throughput through the use of OpenFlow protocol with a centralized controller in SDN network. A delay reduction of about 15% is achieved, while a slight improvement of about 1.7 Mbps in throughput is achieved. The work also showed that how the SDN simplified the use of controller device to program a module that makes the switches act as Load balancer besides the usual forwarding. This is demonstrated here for small network and can be expanded to larger network.

References

i. G. Shainer, "SDN Introduces Choice To The Data Center", Dec. 5, 2015, http://www.networkcomputing.com/data-centers/sdn-introduces-choice-data-center/31944585, last accessed Sept. 2016.

ii. W. Braun, and M. Menth, "Software Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices", University of Tuebingen, Germany, 12 May 2014.

iii. F. Keti and S. Askar, "The Emulation of SDN using Mininet in Different Simulation Environments", Intelligent Systems, Modelling and Simulation (ISMS), IEEE 2015 6th International Conference, 29th October 2015.

iv. M. Koerner, and O. Kao, "Multiple Service Load Balancing with OpenFlow", IEEE 13th International Conference on High Performace Switching and Routing, University of Berlin, Berlin, 2012.

v. N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN: an intellectual history of programmable networks", ACM SIGCOMM Computer Communication Review, New York, USA, April 2014.

vi. X. Foukas, M. Marina, K. Kontovasilis, "Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture", 1st Edition, WIELY Publisher, Finland, March 2015.

vii. K. Salchow, "Load balancing 101: Nuts and Bolts", Technical Marketing and Syndication, April 23, 2012.

viii. D. Kreutz, F. Ramos, P. Verissimo, C. Esteve, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey", Proceedings of the IEEE. 2015 Jan.

ix. V. Shukla, "Introduction to Software Defined Networking-Openflow & VxLAN", 1st Edition, CreateSpace Independent Publishing Platform, June 18, 2013.

x. S. Seo, "Dynamic Traffic Engineering for Improving Energy Efficiency and Network Utilization of Data Center Networks", Pohang University, Korea, Dec. 20, 2013.

xi. P. Goransson and C. Black, "Software Defined Networks A comprehensive Approach", 1st Edition, Morgan Kaufmann Publishers, June 2014.

xii. G. Shainer, "SDN Introduces Choice To The Data Center", Dec. 5, 2015, http://www.networkcomputing.com/data-centers/sdn-introduces-choice-data-center/31944585, last accessed Sept. 2016.

xiii. Cisco Systems and SDx Central, "SDN Controllers Report", 2015, last accessed Sept. 2016, https://www.sdxcentral.com/sdn/definitions/sdn-controllers/.

xiv. P. Sparrow, "Tree topology advantages and disadvantages", 2012, last accessed July 18, 2016, http://www.ianswer4u.com/2012/01/tree-topology-advantages and.html#ixzz4DE7erp64.

xv. B. Lantz, "Mininet and Open vSwitch", Open vSwitch Fall Conference, Open Networking Laboratory, November 16, 2015.

xvi. M. Basheer, E. Bassam "Survey of software components to emulate OpenFlow protocol as an SDN implementation", Dept. of Electrical Engineering, Mosul University, Mosul, Iraq, Science Publishing Group, American Journal of Software Engineering and Applications. Vol. 3, No. 6, 2014, pp. 74-82, December 2014.

xvii. S. Amir, "Software Defined Network Based on OpenFlow Protocol", M.Sc. Thesis Submitted to the College of Information Engineering, University of Al-Nahrain, Iraq, September 2016.

xviii. N. Gonçalves, "A Testbed for research and development of SDN applications using OpenFlow", Telecommunications and Informatics Engineering, Técnico Lisboa, June 2014.

xix. M. McCauley, and A. Al-Shabibi, "POX Wiki", March 2015, last accessed 22 July 2016, https://openflow.stanford.edu/display/ONL/POX+Wiki.

xx. Open Networking Foundation, "OpenFlow Switch Specification version 1.0.0", ONF TS-01, Des. 31, 2009.